12

GEC/ISP/TR-82-388200-6

AN EMPIRICAL EVALUATION OF LANGUAGE-TAILORED PDLS

DEBORAH A. BOEHM-DAVIS
SYLVIA B. SHEPPARD
JOHN W. BAILEY
ELIZABETH KRUESI

Software Management Research
Information Systems Programs
General Electric Company
1755 Jefferson Davis Highway
Arlington, Virginia 22202

MAY 1982

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A115 632	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Empirical Evaluation of Language-Tailored PDLs		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER GEC/ISP/TR-82-388200-6
7. AUTHOR(s) Deborah A. Boehm-Davis, Sylvia B. Sheppard, John W. Bailey, & Elizabeth Kruesi		8. CONTRACT OR GRANT NUMBER(s) N00014-79-C-0595
9. PERFORMING ORGANIZATION NAME AND ADDRESS Information Systems Programs General Electric Company 1755 Jefferson Davis Hwy., Arlington, VA 22202		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N 42 RR04209 01 NR 196-160
11. CONTROLLING OFFICE NAME AND ADDRESS Engineering Psychology Group, Code 442 Office of Naval Research Arlington, VA 22217		12. REPORT DATE May 1982
		13. NUMBER OF PAGES 50
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) SAME		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) SAME		
18. SUPPLEMENTARY NOTES Technical Monitor: Dr. John J. O'Hare		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Software engineering, Software experiments, Structured programming, Modern programming practices, Software documentation, Flowcharts, Program design language, Software human factors, Software specifications.		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Recent research in the area of program documentation has demonstrated a superiority for coding done with a detailed design written in a Program Design Language (PDL) over other formats such as flowcharts. Because PDL is more code-like than other formats, there is less translation required in mapping from the design to the code. If the amount of translation is a critical underlying factor, the optimal PDL for any given implementation will be one

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

that is tailored toward the particular language being used. This experiment evaluated the effectiveness of using a PDL specifically designed to aid in coding the corresponding programming language. This was done by designing PDLs which reflected the syntax and features of particular programming languages and by examining the performance of programmers coding from these various PDLs in one of two implementation languages.

The participants were presented with three programs, in either FORTRAN or MACRO-11 (PDP-11 assembly language), from which several lines had been deleted. The task was to complete the code on line. For each program, the participants received one version of the PDLs, a listing of the partially completed code and a data dictionary. An interactive data collection system captured the overall time to code and debug the programs and the number and types of errors made. Data on the participants' previous programming experience and subjective ratings of the usefulness of the various forms of PDL were collected from questionnaires completed at the end of the experimental session.

The results showed that (a) it took longer to code programs in MACRO-11 than in FORTRAN, (b) the shortest time to code the programs occurred when the coding language of the PDL matched the actual coding language, and (c) the type of PDL and coding language did not significantly affect the number of errors made in coding while the type of problem did. The data suggest that programmers produce code most quickly from a form of documentation that is closest to the code.

This research suggests that providing detailed design information in terms of a language-specific PDL will lead to a shorter coding period than when a language-independent PDL is used.

GEC/ISP/TR-82-388200-6

AN EMPIRICAL EVALUATION OF LANGUAGE-TAILORED PDLS

DEBORAH A. BOEHM-DAVIS
SYLVIA B. SHEPPARD
JOHN W. BAILEY
ELIZABETH KRUESI

Software Management Research
Information Systems Programs
General Electric Company
1755 Jefferson Davis Highway
Arlington, Virginia 22202

Submitted to:

Office of Naval Research
Engineering Psychology Group
Arlington, Virginia

Contract: N00014-79-C-0595
Work Unit: NR 196-160

MAY 1982

TABLE OF CONTENTS

INTRODUCTION

The means by which design information is communicated among software personnel is an important issue in the development of software. Major tasks in the software life cycle, such as design, coding, testing, and maintenance, are frequently performed by different individuals. Leintz and Swanson (1979) found that typically only about half of a software system's maintenance personnel had been involved in its development. Communication among these personnel is no better than the documentation they develop. Poor documentation techniques can dramatically increase labor costs throughout the labor-intensive software life cycle by making both development and maintenance tasks more difficult. Design information, in particular, must be communicated effectively so that the integrity of a system will not be compromised when modifications are implemented.

The transmission of design information in documentation was examined in an earlier study. Sheppard and Kruesi (1981) compared the performance of programmers who were coding from a number of different documentation formats. A total of nine different formats represented the factorial combination of three types of symbology with three spatial arrangements. These two dimensions were chosen because they are the primary dimensions for categorizing the way in which available documentation aids configure the information they present to programmers (Jones, 1979). The three types of symbology in which information was presented consisted of normal English, program design language, and ideograms. The spatial arrangements of the information used in this experiment were sequential, branching, and hierarchical.

In this experiment, 36 professional programmers were presented with documentation for three programs. Working from this documentation, the participants constructed a section of code at the middle of each program. These sections contained about 15 lines and included the most complex decision structures present in the programs. The difficulty of the coding task was measured by four dependent variables: (1) the time to code and debug, (2) the number of submissions required for a correct run, (3) the number of errors, and (4) the number of editor transactions.

The participants were given a short preliminary exercise to familiarize them with the experimental task. Following the exercise, they were given a program listing, a documentation format, and a data dictionary for each of three modular-sized FORTRAN programs (about 50 lines of code). Across the three programs, they saw each type of symbology and each spatial arrangement. A participant, for example, might see the first program presented in sequential normal English, the second program in hierarchical PDL, and the third program in branching ideograms. Using a text editor, they were asked to code the missing segment of the program at a CRT terminal. An on-line data-collection system recorded all interactions with the editor and the time required for each interaction. An automatic-checking procedure informed the participants if the program had been compiled and run successfully or requested that they continue working until a successful execution had been achieved.

The participants were professional programmers employed by General Electric Company with a mean of 6.3 years of professional programming experience. Across the participants, each program, symbology, and arrangement was presented first, second, and third an equal number of times. At the completion of each experiment, the participants were given a questionnaire asking their preferences regarding the documentation formats they had seen.

A comparison of the individual formats revealed that the constrained language presented in the sequential arrangement (normal PDL) resulted in the highest level of performance as measured by all of the dependent variables. In terms of the mean time required to code and debug the program, the normal PDL required only 16.5 minutes as compared with the 31.8 minutes required for the normal English (sequential natural language). A normal flowchart arrangement (branching ideograms) required an intermediate amount of time, 24.7 minutes. A particularly striking result is that the majority of participants made no errors with the normal PDL. The results from this experiment provide clear evidence that PDL is the optimal documentation format for coding.

The question that arises is why this form of documentation is superior to the other formats tested. The most probable explanation of this superiority is that PDL was the most code-like of all documentation formats tested. As a result, there was less translation required in mapping between the documentation and the code. It is important to note that the participants in these experiments were coding in FORTRAN and that they were given a FORTRAN-like PDL. If the amount of translation is a critical underlying factor, no single form of PDL will be optimal for all coding languages. Rather, the optimal PDL will be one that is tailored toward the particular coding language.

The current research examined this hypothesis by creating and evaluating several different PDLs, each of which was tailored toward a particular coding language.

Alan Perlis (1981, p. 104) has described four classes or levels of languages in common use today based on their power for describing computations. The categorization he has proposed is:

(1) machine assembly language;

(2a) ALGOL-like, such as ALGOL 60, FORTRAN, COBOL, and Pascal;

(2b) ALGOL-like with tasking such as JOVIAL, ALGOL 68, CMS-2, PL/I, and Ada; and

(3) interpretive languages which operate on data structures in parallel such as APL and LISP.

Perlis hypothesizes that the use of a language at a higher level will decrease overall software life-cycle costs by making testing and maintenance easier.

For this experiment, we selected one coding language from each of his three major levels (MACRO-11, FORTRAN, and APL), and designed a PDL tailored to each language. Each PDL was designed such that it resembled its corresponding language while remaining comprehensible to a programmer not skilled in the target language.

Using these materials, we were able to test the hypothesis that the correspondence between the PDL and the coding language is an important determinant of coding performance.

METHOD

Participants

Twenty-four professional programmers from three different locations participated in this experiment. All were General Electric employees. The participants averaged 6.1 (s.d. = 3.7) years of programming experience and had used an average of 6.5 (s.d. = 2.6) programming languages.

Independent Variables

The experiment was designed to study the effects of three independent variables: coding language, type of PDL, and type of problem.

Coding Language. Two coding languages were used in this experiment: MACRO-11 and FORTRAN. MACRO-11 is the machine assembly language for the PDP-11 and represents the lowest class of programming languages as described by Perlis (1981). FORTRAN was chosen as representative of a higher-level language from Perlis' categorization scheme.

Program Design Language. The statements from each program were translated into PDLs which were tailored to each of three coding languages: MACRO-11, FORTRAN, and APL. Each PDL was designed to resemble its corresponding language, but still be comprehensible to a programmer not skilled in the target language. The MACRO PDL used left-handed arrows ($\longleftarrow$) to indicate assignment to registers and variables. Mathematical symbols (e.g., +, -, *) were used to indicate operations on the data. The FORTRAN PDL used the form "set x = " to indicate assignment and used mathematical symbols to indicate operations. Selection and repetition constructs were indented to show the program structure. The APL PDL used mathematical notation to indicate both assignment and operations on the data. Since APL is a vector-oriented language, summation over a range (e.g., $\sum X_i$, for $i = 1, \ldots, n$) was used to indicate operations repeated for each data element. No indentation was used in the APL PDL. An example of each version of PDL is shown for each of the three programs in Appendix A.

Problem. In our previous research (Sheppard, Curtis, Milliman, & Love, 1979), significant differences in programmer performance were often associated with differences among problems. Three problems of varying types were chosen for use in this experiment. A program which simulated the path of a rocket was chosen as representative of an engineering problem. A sorting procedure represented the class of programs that manipulate strings or data objects. A third program, representative of a statistical procedure, calculated a correlation coefficient.

These three programs were based on problems contained in Barrodale, Roberts, and Ehle (1971). The problems were coded in both MACRO-11 and FORTRAN and verified for correctness. Each of the resulting MACRO-11 programs contained approximately 45 lines of executable code while each of the resulting FORTRAN programs contained approximately 30 lines of executable code. In addition, a problem to calculate the greatest common divisor of two numbers was coded in each language and used as a practice program.

A section of 3-6 lines of code was deleted from each program. This section, to be completed by the participants, was located somewhere near the middle of the program. The portions deleted from the MACRO-11 and FORTRAN versions of the same problem were chosen to represent a roughly equivalent number of keystrokes and the same (or similar) functions. The statements which the participants were required to construct consisted of assignment, selection, and iteration statements. All dimension, format, and input-output statements as well as all variable declarations were included in the participants' listings. The three problems are presented in each language in Appendix B. The programs are shown as they were presented to the participants, i.e., with the to-be-completed section deleted. For the reader's convenience, the deleted portions of the programs are presented at the end of each program, enclosed in brackets.

Procedure

Prior to the experiment, the participants were given a 20-minute training session in which they were shown examples of each type of PDL. The experimenter also described the procedure for using the text editor to construct the programs during this session.

Experimental sessions were conducted at CRT terminals on a VAX 11/780. Each participant coded all of the problems in either MACRO-11 or FORTRAN. The participants were first given a practice program from which a single line had been deleted. Identical listings of the code appeared on the CRT screen and on a paper printout. The participants were instructed to complete the code, using the text editor. When satisfied that the program would perform correctly, the participants exited from the editor and activated a command file to compile and run the program. If the compilation or assembly was unsuccessful, a message appeared on the screen directly below the line or lines containing the error. If the program had assembled or compiled, the output from the program appeared on the screen with one of the following messages: "OUTPUT IS CORRECT" or "OUTPUT IS INCORRECT." In the latter case, the participant was asked to correct the errors and submit the run again.

Following the practice program the three experimental programs were presented. For each program, the participants received one version of the PDL. In addition, the participants received identical listings of the partially-completed code on the CRT screen and on a paper printout. They also received a data dictionary containing the variable names, a natural-language description of the variables, and the data types. Across the three programs, each participant saw each type of PDL (MACRO-11, FORTRAN, APL) and each problem (correlation, rocket, sort).

An interactive data collection system prompted the participant throughout the experimental procedure. The system recorded each call for an editor command (i.e., ADD, DELETE, LIST, or CHANGE) and the resultant changes in the program. An interval timer, accurate to the nearest second, recorded the time for each of these actions. When a participant required more than one editing session to complete the program correctly, the experimental system recorded exits from the editor, any compilation errors, and the incorrect outputs generated. From these data, the time to code and debug the programs was calculated by summing the times from the individual editing sessions; time for compiling and running the programs was not included.

The participants spent approximately 14 minutes on each experimental program. They were required to continue working on a program until it was completed successfully. They were allowed to take breaks between programs.

Following the experiment, the participants completed a questionnaire about their previous programming experience. The information requested included number of years of experience, and number of programming languages known. The participants were also asked to rate how easy or hard each PDL was to use and how much they relied on each.

Design

The experimental design used in this experiment was a 2 X 3 X 3 mixed between/within subjects design where the kind of programming language (MACRO-11 or FORTRAN) was a between-subject variable and the kind of PDL (MACRO-11, FORTRAN, or APL) and kind of problem (correlation, sort, or rocket simulation) were within-subject variables. Each individual within a group coded three of the nine possible combinations of PDL and problem in one programming language. For example, a participant in the MACRO-11 group might code the rocket problem working from the MACRO-11 PDL, the correlation problem from the APL PDL, and the sort problem from the FORTRAN PDL. The order in which the participants were observed under each treatment condition was randomized independently for each participant. The analysis of this design was based on an example given in Winer (1971, p. 727-736).

RESULTS

Time to Code and Debug

The participants required an average of 14 minutes to code and debug a program. This represents the amount of time spent studying the program, coding the program, and using the text editor (i.e., the total time spent at the terminal less the time for compiling or assembling, linking, and running).

	PROBLEM		
	CORRELATION	ROCKET	SORT
MEAN TIME TO COMPLETE CODING TASK *(MINUTES)*	10.4	14.0	18.8
MEAN NUMBER OF ERRORS	0.3	1.0	1.5

Table 1.
A Comparison of the Dependent Variables for the Three Algorithms

The mean times to complete the code for each of the three programs is shown in Table 1. As can be seen from the table, there were large differences in the amount of time required to complete the programs. The correlation problem required the least amount of time to complete (10.4 minutes), while the sort problem required the greatest amount of time (18.8 minutes). An analysis of variance on the coding times across problems supported this conclusion ($\underline{F}$ (2,36) = 9.29, $\underline{p} < .01$, MS_e = 46.42).

Table 2 shows the mean times for each combination of PDL and coding language. (The shaded portions in this, and subsequent, tables indicate the conditions where the best performance was expected on the basis of a match between actual coding and PDL coding language.) For those participants coding in FORTRAN, the problem coded using the FORTRAN PDL required the least amount of time (6.7 minutes). For those participants coding in MACRO-11, the problem coded using the MACRO-11 PDL required the least amount of time (12.2 minutes). Regardless of coding language, the problem coded using the APL PDL required the most time to code (Mean = 19.4 minutes).

CODING LANGUAGE	PDL		
	MACRO—11	FORTRAN	APL
MACRO—11	12.2	17.7	21.2
FORTRAN	11.3	6.7	17.5

Table 2. Mean Time to Complete Coding Task (in Minutes)

An analysis of variance on the coding times revealed that there was a main effect for coding language (*F* (1,18) = 7.00, *p* .01, MS_e = 70.31) and for PDL (*F* (2,36) = 9.58, *p* .01, MS_e = 46.42). In general, coding in FORTRAN took less time than coding in MACRO-11; it took about the same amount of time to code from the MACRO-11 and FORTRAN PDLs and it took longer to code from the APL PDL.

In addition, the interactions between PDL and coding language (*F* (2,36) = 3.53), PDL and problem (*F* (2,36) = 4.21), and PDL, problem, and coding language (*F* (2,36) = 4.87) were all significant at the .05 level. An examination of the data revealed that the interaction between PDL and coding language arose from the fact that the MACRO-11 PDL led to the shortest time when coding in MACRO-11 while the FORTRAN PDL led to the shortest time when coding in FORTRAN. The interaction between PDL and problem arose from the fact that the time required to code the problems increased from the MACRO-11 PDL through the FORTRAN PDL to the APL PDL when coding the correlation and sort problems. On the other hand, the time required for the MACRO-11 and APL PDLs was roughly equal and they both required more time than the FORTRAN PDL when coding the rocket problem. The underlying cause of this interaction and of the three-way interaction is unclear.

Errors

For programs that did not run successfully on the first submission, the participants' editing activities for subsequent submissions were analyzed to determine the number of errors. Tables 1 and 3 show that the number of errors was very low. Due to the low number of errors, no categorization of the different types of errors was carried out. An analysis of variance on the error data showed a significant effect due to problem (*F* (2,36) = 4.78, $p < .05$, $MS_e = 1.72$), with the sort program resulting in the greatest number of errors and the correlation program in the fewest number. None of the other main effects or interactions were significant at the .05 level. However, the errors do show the same trend as the coding times. The smallest number of errors for each coding language was associated with the PDL tailored to that coding language. In addition, the APL PDL was associated with the largest number of errors for both coding languages.

CODING LANGUAGE	PDL		
	MACRO—11	FORTRAN	APL
MACRO—11	0.6	1.3	1.7
FORTRAN	0.4	0.3	1.2

Table 3. Mean Number of Errors

Preferences for PDL

Across the three programs, the participants received a PDL tailored toward each of the three coding languages. On the questionnaire, they were asked to state which PDL was the easiest to use and which was the hardest to use. They were also asked to rate how much they relied on each version of PDL on a seven-point scale (from 0 = not at all to 7 = constantly throughout). Table 4 shows the number of people in each coding language condition who chose each PDL as the easiest to use while Table 5 shows the number of people who chose each PDL as the hardest to use. (One participant in the FORTRAN coding condition said that all three versions of PDL were the easiest to use; that response was not included in the tallies shown in the table.)

CODING LANGUAGE	PDL		
	MACRO—11	FORTRAN	APL
MACRO—11	9	2	1
FORTRAN	1	10	0

Table 4. Number of Times PDL Chosen as Easiest to Use

CODING LANGUAGE	PDL		
	MACRO—11	FORTRAN	APL
MACRO—11	1	1	10
FORTRAN	9	0	2

Table 5. Number of Times PDL Chosen as Hardest to Use

It can be seen that participants coding in MACRO-11 found the MACRO-11 PDL the easiest to use and the APL PDL the hardest to use. On the other hand, participants coding in FORTRAN found the FORTRAN PDL the easiest to use and the MACRO-11 PDL (rather than the APL PDL) the hardest to use. Table 6 shows the mean rating of how much they relied on each PDL for each problem in each coding language condition.

CODING LANGUAGE	PDL	PROBLEM			
		CORRELATION	ROCKET	SORT	TOTAL
MACRO-11	MACRO-11	5.8	5.5	5.8	5.7
	FORTRAN	4.5	4.0	3.8	4.1
	APL	3.3	3.3	0.5	2.3
FORTRAN	MACRO-11	3.8	5.0	5.3	4.7
	FORTRAN	5.0	4.5	4.8	4.8
	APL	4.3	5.5	1.8	3.8

Table 6. Mean Ratings of Reliance Upon Each PDL

(Scale: 0 = not at all, 7 = constantly throughout)

Participants coding in MACRO-11 stated that they relied most heavily on the MACRO-11 PDL, followed by the FORTRAN and APL PDLs, respectively. Overall, participants coding in FORTRAN stated that they relied most heavily on the FORTRAN PDL, followed by the MACRO-11 and APL PDLs, respectively. However, a closer examination of the table reveals that for the correlation and rocket programs, the participants relied about equally upon the MACRO-11 and APL PDLs. It was only for the sort program that the participants relied less on the APL PDL than on the MACRO-11 PDL.

Experiential Factors

The participants were asked the number of years they had programmed professionally and the number of programming languages they knew. No correlation was found between time to code and debug and these experiential factors.

DISCUSSION

Substantial differences were observed among the three problems used in this experiment. The correlation problem was associated with the shortest times and fewest errors, the sort problem resulted in the poorest performance, and the rocket problem was in-between. This result parallels our past experiences in finding substantial differences across problems.

Substantial differences were also observed among the three versions of PDL. For programmers coding in MACRO-11, the MACRO-11 PDL was associated with the shortest coding times. For participants coding in FORTRAN, the FORTRAN PDL led to the shortest coding times. This result suggests that it is easiest to code in a PDL tailored to your particular coding language.

For participants coding in both FORTRAN and MACRO-11, the APL PDL was associated with the longest coding times. The differences among the three programming languages explain this result. APL is an extremely concise programming language, requiring fewer instructions than either FORTRAN or MACRO-11. This difference among the languages is reflected in the corresponding PDLs. Thus the participants were required to contribute more of the details required for coding when using the APL PDL than when using either the FORTRAN or MACRO-11 PDLs for programming in those two languages. The lack of detail in the APL PDL is probably responsible for the increase in coding times.

Although no significant differences in the number of errors were found among the three versions of PDL for participants coding in either language, the errors did follow the same trend as the coding times. The smallest number of errors for each coding language was associated with the PDL tailored to that coding language. In addition, for each coding language, the APL PDL was associated with the largest number of errors. The overall number of errors was quite low. This is probably due to the fact that the number of lines to be coded was small. If the tasks had been longer, it is possible that significant differences as a function of type of PDL would have been found.

The participants' choices for the easiest/hardest to use PDL and ratings of how much they relied on each PDL provided very clear results. Participants coding in MACRO-11 found the MACRO-11 PDL easiest to use, and participants coding in FORTRAN found the FORTRAN PDL easiest to use. This result parallels the results for the coding times and supports the notion that ease of translation is an important determinant of performance.

Participants coding in MACRO-11 said they found the APL PDL hardest to use and this was reflected in their performance. These participants took longer to produce the missing code when they were using the APL PDL (21.2 minutes) than when they were using the MACRO-11 (12.2 minutes) or FORTRAN (17.7 minutes) PDLs. However, the participants coding in FORTRAN said they found the MACRO-11 PDL hardest to use in spite of the fact that they did better with the MACRO-11 PDL than with the APL PDL. These participants coded the programs considerably more quickly when using the MACRO-11 PDL (11.2 minutes) than when using the APL PDL (17.5 minutes). One explanation for this result may be the difference in level of detail of the three PDLs. The APL PDL contains high-level concepts, the FORTRAN is in-between, and the MACRO-11 contains low-level details. The high-level concepts of APL (e.g., "Sort X_i such that $X_i \leqslant X_{(i+1)}$") may not provide enough detail to be useful to the MACRO-11 or FORTRAN programmers in actually producing code. On the other hand, the MACRO-11 PDL provides low-level details, such as which values are stored in which registers, which is more information than a programmer would require to code in FORTRAN. This suggests that while code cannot always be derived directly from an APL PDL, it can be from a MACRO-11 PDL, although the programmer is required to interpret and integrate several lines of MACRO-11 PDL to obtain one line of FORTRAN code. The effort expended in translating the MACRO-11 PDL into FORTRAN statements may be what is reflected in the choice of the MACRO-11 PDL as the hardest to use for the participants coding in FORTRAN.

The ratings by the participants coding in FORTRAN of how much they relied on each PDL would tend to support this explanation. Overall, these participants' ratings suggest that they relied more heavily upon the MACRO-11 than on the APL PDL (see Table 6). A closer examination of the ratings reveals that for the correlation and rocket problems, the participants relied about equally on the MACRO-11 and APL PDLs; however, for the sort problem, the participants relied much more heavily on the MACRO-11 PDL than on the APL PDL. An examination of the PDLs for each problem suggests that the APL PDL for the sort problem is the most succinct, and provides the least amount of guidance to a programmer as to the content of the actual code.

As in our previous experiments, we compared performance to several experiential factors. In some of our previous experiments, number of programming languages known was highly correlated with performance, while years of programming experience was not correlated with performance. In this experiment, years of experience and number of programming languages known were not correlated with each other or with performance. This difference from past results may be explained by the smaller sample size used in this experiment.

Taken as a whole, the data from this experiment suggest that providing detailed design information in terms of a language-specific PDL will lead to a shorter coding period than when a language-independent PDL is used.

1.00 TITLE PROGRAM CORRELATION

ACKNOWLEDGEMENTS

The authors would like to thank Sue Hannon and John McBeth of GE in Lanham, Maryland, Dave Markham of GE in Sunnyvale, California, and Roger Collins of GE in Arlington, Virginia for providing participants and facilities; Bryan Wolfe for constructing the MACRO-11 programs; Dave Morris and Pete McEvoy for designing the automatic data collection system; Dr. John O'Hare for advice; and Tom McDonald for preparing materials and statistical analyses.

REFERENCES

Barrodale, I., Roberts, F. D. K., & Ehle, B. L. Elementary computer applications in science, engineering, and business. New York: Wiley, 1971.

Jones, C. A survey of programming design and specification techniques. In Proceedings of the IEEE Conference on Specifications of Reliable Software. New York: IEEE, 1979.

Leintz, B. P. & Swanson, E. G. The use of productivity aids in system development and maintenance (Technical Report 79-1). Los Angeles: UCLA, Graduate School of Management, 1979.

Perlis, A. J. Controlling software development through the life cycle model. In A. J. Perlis, F. G. Sayward, & M. Shaw (Eds.), Software Metrics: An analysis and evaluation. Cambridge: The MIT Press, 1981.

Sheppard, S. B., Curtis, B., Milliman, P., & Love, T. Modern coding practices and programmer performance. Computer, 1979, 12 (12), 41-49.

Sheppard, S. B. & Kruesi, E. The effects of the symbology and spatial arrangement of software specifications in a coding task. In Proceedings of Trends & Applications 1981: Advances in Software Technology. New York: IEEE, 1981.

Winer, B. J. Statistical principles in experimental design. New York: McGraw-Hill, 1971.

APPENDIX A

PDL FORMATS

```
PROGRAM CORRELATION
MAIN:  CALL CORR1                     ;READS N, ARRAY X AND ARRAY Y
       SUMX←Ø
       SUMY←Ø
       SUMXY←Ø
       SUMXSQ←Ø
       SUMYSQ←Ø
       RØ←N
       RØ←2 * RØ
       R1←2
LOOP:  SUMX←SUMX + X(R1)
       SUMY←SUMY + Y(R1)
       SUMXSQ←SUMXSQ + X(R1) * X(R1)
       SUMYSQ←SUMYSQ + Y(R1) * Y(R1)
       SUMXY←SUMXY + X(R1) * Y(R1)
       R1←R1 + 2
       IF R1≤RØ GO TO LOOP
       SET FLOAT MODE
       SET (SHORT) INTEGER
       AC2←SUMXSQ - (SUMX * SUMX) ÷ N
       AC3←SUMYSQ - (SUMY * SUMY) ÷ N
       PARM←AC2 * AC3
       R5←↑PBLK
       CNUM←SUMXY - (SUMX * SUMY) ÷ N
       CALL $SQRT
       CDEN←RØ
       CDEN + 2←R1
       CORR←CNUM ÷ CDEN
       CALL CORR2                     ;WRITES CORR
       END OF CORR
```

PRECEDING PAGE BLANK-NOT FILMED

```
PROGRAM CORR
READ FROM 'RDATA': N
FOR I FROM 1 TO N DO
  READ FROM 'RDATA': X(I)
  READ FROM 'RDATA': Y(I)
ENDDO
SET SUMX = 0
SET SUMY = 0
SET SUMXSQ = 0
SET SUMYSQ = 0
SET SUMXY = 0
FOR I FROM 1 TO N DO
  SET SUMX = SUMX + X(I)
  SET SUMY = SUMY + Y(I)
  SET SUMXSQ = SUMXSQ + X(I)**2
  SET SUMYSQ = SUMYSQ + Y(I)**2
  SET SUMXY = SUMXY + X(I) * Y(I)
ENDDO
SET CORR = (SUMXY - SUMX * SUMY/N)/
        SQRT((SUMXSQ - SUMX**2/N) *
        (SUMYSQ - SUMY**2/N))
PRINT CORR
END OF CORR
```

```
PROGRAM CORR
READ $X_i$                    FOR $i = 1,\dots,N$
READ $Y_i$                    FOR $i = 1,\dots,N$
```

$$R = \frac{\left(\sum X_i Y_i - \left(\sum X_i \sum Y_i\right) \div N\right)}{\left(\left(\sum X_i^2 - \left(\sum X_i\right)^2 \div N\right)\left(\sum Y_i^2 - \left(\sum Y_i\right)^2 \div N\right)\right)^{\frac{1}{2}}}$$

```
PRINT R
END OF CORR
```

```
PROGRAM ROCKET
MAIN:  CALL ROCK1      ;READS IN 3 INTEGERS: ACC, MAXTIM, TSTEP
       R4 ← MAXTIM/TSTEP
       N ← R4
       R4 ← (R4 + 1) * 2
       R1 ← 2
LOOP:  T(R1) ← ((R1 - 2)/2) * TSTEP
       VEL(R1) ← ACC * T(R1)
       SUMVEL ← Ø
       R2 ← 2
INNER:  SUMVEL ← SUMVEL + VEL(R2)
        R2 ← R2 + 2
        IF R2 ≤ R1 GO TO INNER
        DIST(R1) ← SUMVEL * TSTEP
        R1 ← R1 + 2
        IF R1 ≤ R4 GO TO LOOP
        CALL ROCK2     ;WRITES T, DIST, VEL
END OF ROCKET
```

```
PROGRAM ROCKET
READ FROM 'RDATA':  ACC, MAXTIM, TSTEP
SET N = MAXTIM/TSTEP
FOR I FROM 1 TO N + 1 DO
  SET T(I) = TSTEP * (I - 1)
  SET VEL(I) = T(I) * ACC
  SET SUMVEL = 0
  FOR J FROM 1 TO I DO
    SET SUMVEL = SUMVEL + VEL(J)
  ENDDO
  SET DIST(I) = TSTEP * SUMVEL
ENDDO
FOR I FROM 2 TO N + 1 DO
  PRINT T(I), DIST(J), VEL(I)
ENDDO
END OF ROCKET
```

```
PROGRAM ROCKET
READ ACC
READ MAXTIM
READ TSTEP
ACC = FORCE ÷ MASS
N = MAXTIM ÷ TSTEP
$T_i = i-1\,(TSTEP)$                        FOR $i = 1,\ldots,N+1$
$VEL_i = T_i(ACC)$                          FOR $i = 1,\ldots,N$
$DIST_i = TSTEP\left(\sum_{j=1}^{i} VEL_j\right)$     FOR $i = 1,\ldots,N$
PRINT $T_i$                                 FOR $i = 1,\ldots,N$
PRINT $DIST_i$                              FOR $i = 1,\ldots,N$
PRINT $VEL_i$                               FOR $i = 1,\ldots,N$
END OF ROCKET
```

```
PROGRAM SORT
MAIN:   CALL SORT1                      ;READS IN N AND ARRAY X
        R1 ← N
        R1 ← 2 * R1
        DONE ← FALSE
ILOOP:    IF R1 ≤ 2 GO TO SORTDN
          IF DONE GO TO SORTDN
          R2 ← 2
          DONE ← TRUE
JLOOP:    IF R2 ≥ R1 GO TO NEXTI
          IF X(R2) ≤ X + 2(R2) GO TO NEXTJ
          R5 ← X(R2)
          X(R2) ← X + 2(R2)
          X + 2(R2) ← R5
          DONE ← FALSE
NEXTJ:    R2 ← R2 + 2
          GO TO JLOOP
NEXTI:    R1 ← R1 - 2
          GO TO ILOOP
SORTDN:    CALL SORT2                   ;WRITES ARRAY X IN ASCENDING ORDER

END OF SORT
```

```
PROGRAM SORT
READ N
FOR I = 1 TO N
  READ FROM 'SDATA': X(I)
ENDDO
SET I = N
SET DONE = FALSE
WHILE I ≥ 2  OR  DONE = TRUE DO
  SET J = 1
  SET DONE = TRUE
  WHILE J < I DO
    IF (X(J) > X(J+1)) THEN
      SET TEMP = X(J)
      SET X(J) = X(J + 1)
      SET X(J + 1) = TEMP
      SET DONE = FALSE
    ENDIF
    SET J = J + 1
  ENDDO
  SET I = I - 1
ENDDO
PRINT X(I) FOR I = 1,...,N
END OF SORT
```

```
PROGRAM SORT
READ $X_i$                               FOR $i = 1,...,N$
SORT $X_i$ SUCH THAT $X_i \leq X_{(i+1)}$        FOR $i = 1,...,N - 1$
PRINT $X_i$                              FOR $i = 1,...,N$
END OF SORT
```

APPENDIX B

PROGRAM LISTINGS

```
        .TITLE  PROGRAM CORRELATION
        ;THIS PROGRAM CALCULATES A CORRELATION COEFFICIENT FOR
        ;TWO SETS OF NUMBERS
        ; REGISTER AND ACCUMULATOR DESCRIPTIONS:
        ; RO : NUMBER OF PAIRS OF ITEMS, EQUAL TO N
        ; R1 : LOOP COUNTER
        ; R3 : WORKING STORAGE
        ; AC0 : FLOATING POINT REPRESENTATION OF N
        ; AC1 : FLOATING POINT WORKING STORAGE
        ; AC2 : FLOATING POINT WORKING STORAGE
        ; AC3 : FLOATING POINT WORKING STORAGE
        ; REMINDER: THE PDP 11 IS A BYTE-ADDRESSABLE COMPUTER.
        ; THE ADDRESSES OF CONSECUTIVE WORDS OF STORAGE DIFFER
        ; BY TWO.
AC0=%0
AC1=%1
AC2=%2
AC3=%3
        .GLOBL  MAIN
        .PSECT  CDATA,D,GBL,OVR
X:      .BLKW   100.
Y:      .BLKW   100.
N:      .WORD   5
CORR:   .BLKW   2
PBLK:   .WORD   1,PARM
PARM:   .BLKW   2
SUMX:   .BLKW   1
SUMY:   .BLKW   1
SUMXY:  .BLKW   1
SUMXSQ: .BLKW   1
SUMYSQ: .BLKW   1
CNUM:   .BLKW   2
CDEN:   .BLKW   2
        .PSECT  $CODE1,I,CON
MAIN:   CALL    CORR1
        CLR     SUMX
        CLR     SUMY
        CLR     SUMXY
        CLR     SUMXSQ
        CLR     SUMYSQ
        MOV     N,RO
        ASL     RO
        MOV     #2,R1
LOOP:   NOP
        ; ***YOUR CODE GOES HERE***






        MOV     Y(R1),R3
        MUL     R3,R3
        ADD     R3,SUMYSQ
        MOV     X(R1),R3
        MUL     Y(R1),R3
        ADD     R3,SUMXY
        ADD     #2,R1
        CMP     R1,RO
        BLE     LOOP
        SETF
```

PRECEDING PAGE BLANK-NOT FILMED

```
700              SETI
710              LDCIF    SUMX, AC1
720              MULF     AC1, AC1
730              LDCIF    N, AC0
740              DIVF     AC0, AC1
750              LDCIF    SUMXSQ, AC2
760              SUBF     AC1, AC2
770              LDCIF    SUMY, AC1
780              MULF     AC1, AC1
790              DIVF     AC0, AC1
800              LDCIF    SUMYSQ, AC3
810              SUBF     AC1, AC3
820              MULF     AC3, AC2
830              STF      AC2, PARM
840              MOV      #PBLK, R5
850              LDCIF    SUMX, AC1
860              LDCIF    SUMY, AC2
870              MULF     AC2, AC1
880              DIVF     AC0, AC1
890              LDCIF    SUMXY, AC3
900              SUBF     AC1, AC3
910              STF      AC3, CNUM
920              CALL     $SQRT
930              MOV      R0, CDEN
940              MOV      R1, CDEN+2
950              LDF      CNUM, AC0
960              DIVF     CDEN, AC0
970              STF      AC0, CORR
980              CALL     CORR2
990              RETURN
999              .END
```

```
500              ADD      X(R1), SUMX
510              ADD      Y(R1), SUMY
520              MOV      X(R1), R3
530              MUL      R3, R3
540              ADD      R3, SUMXSQ
```

```
100 C       PROGRAM CORRELATION
110 C       THIS PROGRAM CALCULATES THE CORRELATION COEFFICIENT FOR TWO
120 C       SETS OF NUMBERS
130         INTEGER X(100), Y(100)
140         OPEN (UNIT=3, NAME='CDATA', TYPE = 'OLD')
150         READ (3, 1000) N
160  1000   FORMAT (I5)
170         DO 100 I = 1, N
180         READ (3, 1001) X(I)
190   100   CONTINUE
200         DO 110 I = 1, N
210         READ (3, 1001) Y(I)
220   110   CONTINUE
230  1001   FORMAT(I3)
240         SUMX = 0
250         SUMY = 0
260         SUMXY = 0
270         SUMXSQ = 0
280         SUMYSQ = 0
290         DO 300 I = 1, N
300 C          ***YOUR CODE GOES HERE***
310
320
330
340
350
360
500         SUMXY = SUMXY + (X(I) * Y(I))
510   300   CONTINUE
520         CORR = (SUMXY - SUMX * SUMY/N)/
530        1 SQRT((SUMXSQ - SUMX **2/N)*
540        2(SUMYSQ - SUMY ** 2/N))
550         WRITE(6, 1002) CORR
560  1002   FORMAT('    CORR =  ', F16.5)
570         CLOSE(UNIT=3)
580         STOP
590         END
```

```
300         SUMX = SUMX + X(I)
310         SUMY = SUMY + Y(I)
320         SUMXSQ = SUMXSQ + X(I)**2
330         SUMYSQ = SUMYSQ + Y(I)**2
```

```
100                 .TITLE  PROGRAM ROCKET
110                   ;THIS PROGRAM SIMULATES THE PATH OF A ROCKET
120                   ; REGISTER DESCRIPTIONS:
130                   ; R1 :   INDEX INTO ARRAYS T, VEL, AND DIST
140                   ; R2 :   INDEX INTO ARRAY VEL FOR COMPUTING
150                   ;        THE SUM OF THE VELOCITIES
160                   ; R3 : WORKING STORAGE
170                   ; R4, R5 : HOLD THE QUOTIENT AND REMAINDER, RESPECTIVELY,
180                   ;        WHEN MAXTIM/TSTEP IS CALCULATED. R4 THEN BECOMES
190                   ;        THE LOOP COUNTER
200                   ; NOTE. INCLUDING INITIALIZATIONS, N + 1 VALUES
210                   ;        WILL BE COMPUTED.  N IS CONVERTED TO
220                   ;        BYTE COUNT
221                   ; REMINDER: THE PDP 11 IS A BYTE-ADDRESSABLE COMPUTER.
222                   ;        THE ADDRESSES OF CONSECUTIVE WORDS OF STORAGE
223                   ;        DIFFER BY TWO.
230                 .GLOBL  MAIN
240                 .PSECT  RDATA,D,GBL,OVR
250     T:          .BLKW   50.
260     DIST:       .BLKW   50.
270     VEL:        .BLKW   50.
280     ACC:        .BLKW   1.
290     N:          .BLKW   1.
300     MAXTIM:     .BLKW   1.
310     TSTEP:      .BLKW   1.
320     SUMVEL:     .BLKW   1.
330                 .PSECT  $CODE1,I,CON
340     MAIN:       CALL    ROCK1
350                 MOV     MAXTIM,R5
360                 CLR     R4
370                 DIV     TSTEP,R4
380                 MOV     R4,N
390                 INC     R4
400                 ASL     R4
410                 MOV     #2,R1
420     LOOP:       MOV     R1,R3
430                 SUB     #2,R3
440                 ASR     R3
450                 MUL     TSTEP,R3
460                 MOV     R3,T(R1)
470                 MUL     ACC,R3
480                 MOV     R3,VEL(R1)
490                 CLR     SUMVEL
500                 MOV     #2,R2
510     INNER:      ADD     VEL(R2),SUMVEL
520                 ADD     #2,R2
530                 ; ***YOUR CODE GOES HERE***
540
550
560
570
580
590
600                 ADD     #2,R1
610                 CMP     R1,R4
620                 BLE     LOOP
630                 CALL    ROCK2
620                 RTS     PC
650                 .END
```

```
530         CMP         R2,R1
540         BLE         INNER
550         MOV         SUMVEL,R3
560         MUL         TSTEP,R3
570         MOV         R3,DIST(R1)
```

```
100             .TITLE  PROGRAM SORT
110                ;THIS PROGRAM SORTS INTEGERS IN ASCENDING ORDER
120                ; REGISTER DESCRIPTIONS.
130                ; R1 : POINTS TO THE ARRAY ELEMENT
140                ;      IN THE HIGHEST ADDRESS NOT YET SORTED
150                ; R2 : POINTS INTO ARRAY X AND ARRAY SEQUENCES
160                ;      FROM THE ELEMENT IN THE LOWEST ADDRESS
170                ;      UP TO R1
180                ; R5 : TEMPORARY STORAGE FOR SWAPPING ARRAY
190                ;      ELEMENTS
191                ; REMINDER: THE PDP 11 IS A BYTE-ADDRESSABLE COMPUTER.
192                ;      THE ADDRESSES OF CONSECUTIVE WORDS OF STORAGE
193                ;      DIFFER BY TWO.
200             .GLOBL  MAIN
210             .PSECT  SDATA,D,GBL,OVR
220 N:          .BLKW   1
230 X:          .BLKW   100.
240 DONE:       .BLKW   1
250             .PSECT  $CODE1,I,CON
260 MAIN:       CALL    SORT1
270             MOV     N,R1
280             ASL     R1
290             CLR     DONE
300 ILOOP:      CMP     R1,#2
310             BLE     SORTDN
320             TST     DONE
330             BNE     SORTDN
340             MOV     #2,R2
350             MOV     #1,DONE
360 JLOOP:      NOP
370             ; ***YOUR CODE GOES HERE***
380
390
400
410
420
430
500             MOV     X(R2),R5
510             MOV     X+2(R2),X(R2)
520             MOV     R5,X+2(R2)
530             CLR     DONE
540 NEXTJ:      ADD     #2,R2
550             BR      JLOOP
560 NEXTI:      SUB     #2,R1
570             BR      ILOOP
580 SORTDN:     CALL    SORT2
590             RTS     PC
600             .END
```

```
370             CMP     R2,R1
380             BGE     NEXTI
390             CMP     X(R2),X+2(R2)
400             BLE     NEXTJ
```

```
100 C      PROGRAM ROCKET
110 C      THIS PROGRAM SIMULATES THE PATH OF A ROCKET
120        INTEGER MAXTIM, TSTEP, ACC, N, SUMVEL
130        INTEGER  T(50), DIST(50), VEL(50)
140        OPEN (UNIT=3, NAME='PDATA.DAT', TYPE='OLD')
150 1000   FORMAT (3I3)
160        READ (3, 1000) ACC, MAXTIM, TSTEP
170        N = MAXTIM/TSTEP
180        DO 10 I = 1, N+1
190        T(I) =  TSTEP * (I - 1)
200        VEL(I) = T(I) * ACC
210        SUMVEL = 0
220 C      ***YOUR CODE GOES HERE***
230
240
250
260
270
280
400   10   CONTINUE
410   20   WRITE (6, 2000)(T(I), DIST(I), VEL(I), I = 2, N+1)
420 2000   FORMAT ('  T =   ', I10, '  DIST =   ', I10, '  VEL =   ', I10)
430        CLOSE (UNIT=3)
440        STOP
450        END
```

```
220        DO 5  J = 1, I
230    5   SUMVEL = SUMVEL + VEL(J)
240        DIST(I) = TSTEP * SUMVEL
```

```
100 C        PROGRAM SORT
110 C        THIS PROGRAM SORTS INTEGERS IN ASCENDING ORDER
120          INTEGER X(10), N
130          LOGICAL DONE
140          OPEN (UNIT=3,NAME='SDATA.DAT',TYPE='OLD')
150          READ (3,1000) N
160   1000   FORMAT(I3)
170          DO 100 I = 1, N
180          READ(3,1000) X(I)
190    100   CONTINUE
200    200   I = N
210          DONE = FALSE
220    210   IF (I  LT. 2 .OR. DONE )GO TO 300
230          J = 1
240          DONE = TRUE
250    220   CONTINUE
260 C        ***YOUR CODE GOES HERE***
270
280
290
300
310
320
500          TEMP = X(J)
510          X(J) = X(J + 1)
520          X(J + 1) = TEMP
530          DONE = FALSE
540   230    J = J + 1
550          GO TO 220
560   250    I = I - 1
570          GO TO 210
580   300    WRITE (6,2000)(X(I),I = 1, N)
590   2000   FORMAT(10I3)
600          CLOSE(UNIT=3)
610          STOP
620          END
```

```
260          IF (J .GE. I) GO TO 250
270          IF (X(J) .LE. X(J+1)) GO TO 230
```

TECHNICAL REPORTS
DISTRIBUTION LIST

OFFICE OF NAVAL RESEARCH

Code 442

TECHNICAL REPORTS DISTRIBUTION LIST

OSD

Capt. Paul R. Chatelier
Office of the Deputy Under Secretary
of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, DC 20301

Department of the Navy

Engineering Psychology Programs
Code 442
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Communication & Computer Technology
Programs
Code 240
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Tactical Development & Evaluation
Support Programs
Code 230
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Manpower, Personnel and Training
Program
Code 270
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Physiology & Neuro Biology Programs
Code 441 B
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Special Assistant for Marine
Corps Mattrs
Code 100M
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Department of the Navy

Commanding Officer
ONR Eastern/Central Regional Office
ATTN: Dr. J. Lester
495 Summer Street
Boston, MA 02210

Commanding Officer
ONR Western Regional Office
ATTN: Dr. E. Gloye
1030 East Green Street
Pasadena, CA 91106

Office of Naval Research
Scientific Liaison Group
American Embassy, Room A-407
APO San Francisco, CA 96503

Director
Naval Research Laboratory
Technical Information Division
Code 2627
Washington, DC 20375

Dr. Michael Melich
Communications Sciences Division
Code 7500
Naval Research Laboratory
Washington, D.C. 20375

Dr. Louis Chmura
Code 7592
Naval Research Laboratory
Washington, D.C. 20375

Mr. Robert G. Smith
Office of the Chief of Naval
Operations, OP987H
Personnel Logistics Plans
Washington, DC 20350

Dr. Jerry C. Lamb
Combat Control Systems
Naval Underwater Systems Center
Newport, RI 02840

Naval Training Equipment Center
ATTN: Technical Library
Orlando, FL 32813

Department of the Navy

Human Factors Department
Code N-71
Naval Training Equipment Center
Orlando, FL 32813

Dr. Alfred F. Smode
Training Analysis and Evaluation Group
Naval Training Equipment Center
Code TAEG
Orlando, FL 32813

Dr. Albert Colella
Combat Control Systems
Naval Underwater Systems Center
Newport, RI 02840

K.L. Britton
Code 7503
Naval Research Laboratory
Washington, D.C. 20375

Dr. Gary Poock
Operations Research Department
Naval Postgraduate School
Monterey, CA 93940

Dean of Research Administration
Naval Postgraduate School
Monterey, CA 93940

Mr. Warren Lewis
Human Engineering Branch
Code 8231
Naval Ocean Systems Center
San Diego, CA 92152

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, DC 20380

HQS, U.S. Marine Corps
ATTN: CCA40 (Major Pennell)
Washington, D.C. 20380

Department of the Navy

Commanding Officer
MCTSSA
Marine Corps Base
Camp Pendleton, CA 92055

Chief, C3 Division
Development Center
MCDEC
Quantico, VA 22134

Dr. Robert Wisher
Naval Material Command
NAVMAT 0722 - Room 508
800 North Quincy Street
Arlington, VA 22217

Commander
Naval Air Systems Command
Human Factors Programs
NAVAIR 340F
Washington, DC 20361

Commander
Naval Air Systems Command
Crew Station Design
NAVAIR 5313
Washington, DC 20361

Mr. Phillip Andrews
Naval Sea Systems Command
NAVSEA 0341
Washington, DC 20362

Commander
Naval Electronics Systems Command
NC #1 Room 4E56
Code 81323
Washington, DC 20360

Dr. Arthur Bachrach
Behavioral Sciences Department
Naval Medical Research Institute
Bethesda, MD 20014

Dr. George Moeller
Human Factors Engineering Branch
Submarine Medical Research Lab
Naval Submarine Base
Groton, CT 06340

Department of the Navy

Head
Aerospace Psychology Department
Code L5
Naval Aerospace Medical Research Lab
Pensacola, FL 32508

Dr. James McGrath
CINCLANT FLT HQS
Code 04E1
Norfolk, VA 23511

Navy Personnel Research and
Development Center
Planning & Appraisal Division
San Diego, CA 92152

Dr. Robert Blanchard
Navy Personnel Research and
Development Center
Command and Support Systems
San Diego, CA 92152

LCDR Stephen D. Harris
Human Factors Engineering Division
Naval Air Development Center
Warminster, PA 18974

Dr. Julie Hopson
Human Factors Engineering Division
Naval Air Development Center
Warminster, PA 18974

Mr. Jeffrey Grossman
Human Factors Branch
Code 3152
Naval Weapons Center
China Lake, CA 93555

Human Factors Engineering Branch
Code 1226
Pacific Missile Test Center
Point Mugu, CA 93042

Mr. J. Williams
Department of Environmental Sciences
U.S. Naval Academy
Annapolis, MD 21402

Department of the Navy

Dean of the Academic Departments
U.S. Naval Academy
Annapolis, MD 21402

Human Factors Section
Systems Engineering Test Directorate
U.S. Naval Air Test Center
Patuxent River, MD 20670

Dr. Robert Carroll
Office of the Chief of Naval
Operations (OP-115)
Washington, DC 20350

Department of the Army

Mr. J. Barber
HQS, Department of the Army
DAPE-MBR
Washington, DC 20310

Technical Director
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Director, Organizations and Systems
Research Laboratory
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Technical Director
U.S. Army Human Engineering Labs
Aberdeen Proving Ground, MD 21005

ARI Field Unit-USAREUR
ATTN: Library
C/O ODCSPER
HQ USAREUR & 7th Army
APO New York 09403

Department of the Air Force

U.S. Air Force Office of Scientific
Research
Life Sciences Directorate, NL
Bolling Air Force Base
Washington, DC 20332

Department of the Air Force

Chief, Systems Engineering Branch
Human Engineering Division
USAF AMRL/HES
Wright-Patterson AFB, OH 45433

Dr. Earl Alluisi
Chief Scientist
AFHRL/CCN
Brooks AFB, TX 78235

Foreign Addressees

North East London Polytechnic
The Charles Myers Library
Livingstone Road
Stratford
London E15 2LJ
ENGLAND

Professor Dr. Carl Graf Hoyos
Institute for Psychology
Technical University
3000 Munich
Arcisstr 21
FEDERAL REPUBLIC OF GERMANY

Dr. Kenneth Gardner
Applied Psychology Unit
Admiralty Marine Technology
Establishment
Teddington, Middlesex TW11 OLN
ENGLAND

Director, Human Factors Wing
Defence & Civil Institute of
Environmental Medicine
Post Office Box 2000
Downsview, Ontario M3M 3B9
CANADA

Dr. A. D. Baddeley
Director, Applied Psychology Unit
Medical Research Council
15 Chaucer Roa
Cambridge, CB2 2EF
ENGLAND

Other Government Agencies

Defense Technical Information Center
Cameron Station, Building 5
Alexandria, VA 22314 (12)

Other Government Agencies

Dr. Craig Fields
Director, Cybernetics Technology Office
Defense Advanced Research Projects
Agency
1400 Wilson Boulevard
Arlington, VA 22209

Dr. M. Montemerlo
Human Factors & Simulation
Technology, RTE-6
NASA HQS
Washington, D.C. 20546

Other Organizations

Dr. Jesse Orlansky
Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311

Dr. Robert T. Hennessy
NAS - National Research Council
Committee On Human Factors
2101 Constitution Avenue, N.W.
Washington, DC 20418

Dr. Robert Williges
Dept. of Industrial Engineering
Virginia Polytechnical Institute
and State University
130 Whittemore Hall
Blacksburg, VA 24061

Mr. Edward M. Connelly
Performance Measurement
Associates, Inc.
410 Pine Street, S.E.
Suite 300
Vienna, VA 22180

Dr. Richard W. Pew
Information Sciences Division
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02238

ATE

LMED

-8